Design organization : there is more to expert knowledge than is dreamed of in the planner's philosophy

Willemien Visser

▶ **To cite this version:**

Willemien Visser. Design organization : there is more to expert knowledge than is dreamed of in the planner's philosophy. [Research Report] RR-1765, INRIA. 1992. inria-00077005

HAL Id: inria-00077005
https://hal.inria.fr/inria-00077005

Submitted on 29 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1765

Programme 3
Intelligence artificielle, Systèmes cognitifs et Interaction homme-machine

DESIGN ORGANIZATION: THERE IS MORE TO EXPERT KNOWLEDGE THAN IS DREAMED OF IN THE PLANNER'S PHILOSOPHY

Willemien VISSER

Octobre 1992

RR-1765

Programme 3

Communication homme-machine

DESIGN ORGANIZATION: THERE IS MORE TO EXPERT KNOWLEDGE THAN IS DREAMED OF IN THE PLANNER'S PHILOSOPHY

Willemien Visser

July 1992

This paper presents a largely extended version of "Planning and organization in expert design activities". In D. Gilmore, R. Winder & F. Détienne (Eds.), User-centred requirements for software engineering environments. Heidelberg: Springer, to be published.

Design organization:
There is more to expert knowledge
than is dreamed of in the planner's philosophy

Abstract: The organization of actual design activities, even by experts and even in routine tasks, is not appropriately characterized by the retrieval of preexisting plans; this organization is opportunistic. This position is defended through a discussion of an important number of empirical design studies. A major cause why design activities are thus organized is that, even if expert designers possess plans which they may retrieve and use, they very often deviate from these plans in order for their activity to satisfy action-management constraints, the most important being cognitive economy.

Keywords: Planning, Organization, Design activity, Opportunistic organization, Action management, Control, Cognitive cost, Cognitive economy, Expertise, Routine task, Nonroutine task.

L'organisation de la conception:
il y a plus de choses dans l'expertise en conception
qu'il est rêvé dans la philosophie du planificateur

Résumé: La position défendue dans ce texte est la suivante: la récupération en mémoire de plans préexistants ne caractérise pas de façon appropriée l'organisation de l'activité de conception, même pas l'activité d'experts, et même pas l'activité mise en oeuvre dans des tâches routinières. Des résultats d'un nombre important d'études conduites sur des activités de conception sont présentés pour appuyer cette position. L'argument principal provient de résultats d'études empiriques de la conception qui montrent que les activités mises en oeuvre dans des tâches réelles de conception sont organisées de façon opportuniste. Un facteur important qui contribue à cet opportunisme -et sur lequel ce texte se focalise- est l'économie cognitive, un critère utilisé dans la de gestion de l'action. Dû au rôle de ce critère dans la sélection d'actions, des concepteurs - même s'ils possèdent des plans qu'ils peuvent récupérer en mémoire et utiliser- dévient très souvent de leurs plans dans le but de satisfaire ce principal critère au niveau de la gestion de l'action qu'est le coût cognitif.

Mots-clés: Planification, Organisation, Activité de conception, Organisation opportuniste, Gestion de l'action, Contrôle, Coût cognitif, Economie cognitive, Expertise, Tâche routinière, Tâche non-routinière.

Contents

1. INTRODUCTION. PLANS AND ORGANIZATION, ROUTINE TASKS AND EXPERTISE

This paper defends the claim that, even for experts involved in routine design, retrieval of preexisting plans does not appropriately characterize the organization of their actual activity. The arguments that will be used are based on results from empirical studies showing that even the design activities of experts possessing and using plans are opportunistically organized because the designers often prefer to proceed to other actions than those proposed by these plans. This choice is inspired by action-management considerations, the most important being cognitive economy.

In what follows, we first present the distinction between "planning" and "organization" and some hypotheses concerning, on the one hand, the relationship between planning and the routine character of a task, and, on the other hand, the relationship between planning and expertise. Next, section 2 presents and critiques planning models, focusing on schema-based programming models. In section 3, several empirical studies on design in different domains are presented, nearly each one showing one or more factors contributing to the opportunistic character of the organization of the activity. Section 4 presents situational factors underlying the organization of design activities and proposes elements for a model of design which can account for opportunism. Section 5 first presents a recapitulation of the different positions on the relationship between the routine character of a task and the organization of the activity. It then closes with a discussion of some consequences of our conclusions for design support tools.

Planning. In his book "Cognitive psychology of planning", Hoc (1988b) considers a plan to be "a schematic and/or multilevel representation of an object (procedure or state), elaborated and/or used to guide activity".
The anticipatory and schematic characteristics of a plan may guide a subject's activity in at least two ways:

- Declarative plan: Structure of the result of the activity. By showing the structure which the result -or an intermediary state- of the activity must have, a "declarative plan" allows the anticipation of states this activity must attain or go through.

- Procedural plan: Structure of the activity. If a plan represents both the coordination between actions to be realized and elements of the control structure, it may guide the activity in that it provides a schematic representation of these actions and the order in which their execution has to take place.

Planning as one of the component activities of design. Planning can be examined at several levels: as a problem-solving activity with its own components (in the case of plan

construction, see below), or as a component of another global task, such as design. In this text it is considered as a component activity of design, next to components such as the construction of problem representations and the proper problem-solving activities of solution development and solution evaluation (see Visser, 1991b). The function of planning is then to guide, organize and anticipate the actual problem-solving activity, especially concerning the definition, identification and/or choice of subproblems, the order in which these subproblems are to be handled, and the strategies and knowledge sources to be used for solving them.

As soon as a problem-solving activity becomes somewhat complex -such as in real design tasks- a planning component generally appears. If one considers a problem-solving activity, not at the global level, but at the lower level of particular stages, planning may sometimes be absent. When a subproblem, for example, is solved by solution retrieval, the development of this solution may not involve several steps which require organization or anticipation (see Visser, 1991b).

Design may itself be considered as a component of a more global task, such as programming. In this context, "design ... is often considered to be synonymous with planning, i.e. laying out at some level of abstraction, the pieces of the solution and their interrelations" (Pennington & Grabowski, 1990).

Plans and Organization. We distinguish between designers' mental representations underlying their planning and the structure of their actual activity:

- Plan. This notion refers to the mental representations constructed and/or retrieved by designers in order to guide their design activity.

- Organization. This notion will be reserved for the structure of the actual design activity, such as it may be observed by an external observer.

Two forms of planning. Planning activities may roughly take two forms:

- Plan retrieval. The designer may retrieve particular preexisting plans[1] or may instantiate plan schemas. Both are memory representations which result from plan constructions in previous activities. A plan which is retrieved during a problem-solving activity does not necessarily constitute the plan for the entire activity.

---

[1] "Plan" and "subplan" are relative notions (like "problem" and "subproblem"). Except if there is a risk of confusion, we will use the term "plan" for representations which may constitute parts of larger representations.

- Plan construction. Using preexisting plans and other components, new plans may be constructed for an activity. If a plan is constructed, it is rarely constructed entirely before proceeding to the activity that is planned. Planning the problem-solving activity, constructing problem representations, proceeding to solution development and to solution evaluation are not separate components of the activity which are executed consecutively in this -or another- order.

Routine design. A distinction between "routine" (or "familiar") and "nonroutine" (or "creative", "insightful" or "innovative") is commonly used in design studies, but more in the domain of A.I. (see, e.g., Brown & Chandrasekaran, 1989; Navinchandra, 1991) than in cognitive psychology (see, e.g., Kaplan & Simon, 1990; Mayer, 1989).

According to Navinchandra (1991), there is "no objective demarcation between routine and innovative design, but rather a continuum ranging from purely routine to innovative" (p. 7).

His interest is in "innovative conceptual design". "Simply put, something is innovative if it solves a known or a new problem in a way different from other known designs." (p. 3).

"A design process is deemed routine if it involves a well understood sequence of steps where all decision points and outcomes are known a priori - there are no surprises. ... Routine design systems can be organized in several different ways. For example, (1) some routine design tasks can be broken into discrete steps, where each step performs some specific design sub-task; (2) in some cases, the problem can be approached hierarchically, where each level in the hierarchy is predetermined; and (3) in other cases, specific heuristics are available for different parts of the design." (p. 2)

Brown and Chandrasekaran (1989) distinguish three "classes" of design.

Class 1, i.e. "open-ended 'creative' design", "[leads] to major inventions or completely new products". According to the authors, this is "extremely innovative behavior, and [they] suspect that very little design activity is in this class" (p. 33).

Class 2 design is "characterized by the existence of powerful problem decompositions. However, design plans for some of the component problems may be in need of de novo construction or substantial modification" (e.g., design of a new automobile) (ibid.).

Class 3 design, the type studied by the authors, is "relatively routine design [which] ... is not trivial, however.... The design task is still too complex for simple algorithmic solutions or table look up. ... [It still requires] knowledge-based problem-solving." (p. 34)

Notice that these authors suppose that different strategies or processes are involved in "creative" and "routine" design.

Together with several colleagues and students, Simon is studying the processes underlying scientific discovery (see Cheng, 1991; Kulkarni & Simon, 1988; and Qin & Simon, 1990) and the "search of insight" (see Kaplan & Simon, 1990). Kaplan and Simon distinguish "routine" from "insightful" problem solving (focusing themselves on the second type). The authors conclude that "insightful" problem solving does not call for "creative" processes different from those observed in other kinds of problem-solving settings.

Mayer (1989) distinguishes routine and nonroutine problems. "Routine problems are familiar problems that, although not eliciting an automatic memorized answer, can be solved by applying a well-known procedure. Although the problem solver does not immediately know the answer to a routine problem, [they know] how to arrive at an answer. For example, the problem 888 x 888 is a routine problem for most adults. In contrast, nonroutine problems are unfamiliar problems for which the problem solver does not have a well-known solution procedure and must generate a novel procedure." (p. 40)

Comparing these four approaches, one sees that the authors do not agree on whether different types of problems ("routine" or "nonroutine") involve different problem-solving processes. We will adopt the global distinction between "routine" and "nonroutine" problems, as proposed by Mayer, for characterizing the tasks analyzed in this paper.

Note that the "routine" or "nonroutine" character of a design task depends on the designers' knowledge with respect to the "problem" they are confronted with -and possibly on other characteristics of the situation constituted by a task and a subject (see below, §5.2). A "nonroutine" task for one designer may constitute a "routine" task for a colleague. This second designer may retrieve a preexisting procedure leading to the answer or may not even need to proceed to "problem solving" for the execution of the task (when "simple" retrieval of the required answer is sufficient).

The same remark holds for the degree to which designers may be considered "expert" or "novice", because the expertise may be considered to depend on the task they are confronted with. Nevertheless, the characterization of a designer as an "expert" or a "novice" is often used more globally: "expertise" is generally expertise in a problem domain, and not expertise in one particular problem or even type of problem. Designers with great experience in tasks in their domain are generally considered "experts" (in the domain), whereas their colleagues who still have little experience in these tasks are considered to be "novices" (in the domain).

In our discussion of the tasks examined in the studies presented in this paper, we will thus use the distinction between "routine" and "nonroutine". We are not going to question an authors' decision on this point, because the studied tasks and subjects are rarely described in sufficient detail for such questioning. Concepts such as "(moderate) difficulty", "novelty" and "complexity" will be considered synonyms of "nonroutine"; "familiarity" (of a task for a subject) as a synonym of "routine".

Planning and the routine character of the task. At first sight, the following "naïve" hypothesis could be thought to express the relationship between the two forms of planning and routine vs. nonroutine design:

- retrieval of preexisting plans
  would be sufficient for planning routine design tasks;

whereas

- planning in the sense of plan construction
  would be supposed to characterize nonroutine rather than routine design.

Planning and expertise. An analogous hypothesis, establishing a similar relationship between planning and expertise, would be that:

- novices in a domain possess few preexisting plan structures,
  so their planning will tend to be constructive;

whereas

- experts in a domain possess many preexisting plans,
  so they will tend to proceed by plan retrieval.

The data reviewed below impose, however, a revision of these hypotheses, leading to our stand:

even for experts involved in routine design, retrieval of preexisting plans does not appropriately characterize the organization of their activity

N.B. The dimension "routine-nonroutine tasks" may subsume the dimension of "expertise". In the rest of this text, we will continue to use the two dimensions, but the conclusion will relate the organization of the activity implemented in a task to the routine-nonroutine character of this task.

To sum up, in this paper,

- "planning" is analyzed as one of the component activities of the design activity;
- the notion "plan" is used to refer to mental representations designers use in this planning activity, i.e. representations developed for guiding the course of action that they are going to follow (or suppose they are going to follow) in order to construct their design; and
- the notion "organization" refers to the structure of their actual design activity.

2. PLANNING MODELS

The concept of "plan" is central in various models of cognitive activities. In models of activities involving a design component, it is especially used for programming. This section presents briefly the plan-based programming model and a number of critiques it has received. With respect to more general planning models, only a brief presentation will be made of some recent critical reactions upon these models.

Classical, general planning approaches often suppose hierarchical plan structures: e.g., in psychology, Miller, Galanter and Pribram (1960); in A.I., Sacerdoti (1974). Presentations and discussions of these models can be found in many places. Some references are presented below.

Hoc (1988b) provides a cognitive psychology approach to planning, mainly based on results from studies in programming and classical problem-solving tasks.

From a cognitive ergonomics approach, Bisseret (1987/1990) discusses steps towards computer-aided text production in the context of a critical presentation of hierarchical vs. opportunistic planning, and defends his ideas by references to empirical research into design.

Allen, Hendler and Tate (1990) present many of the historically important papers in the domain of A.I. dealing with planning and action.

A historical presentation from an A.I. viewpoint can be found in Chapman (1987). The author describes briefly two important "prehistorical" nonconjunctive planners (GPS by Newell, Shaw and Simon, 1959, and Strips by Fikes and Nilsson, 1971) and then gives a critical chronology of domain-independent conjunctive planners, starting with Sussman's (1973) HACKER and ending with his own system TWEAK.

2.1 The plan-based programming model

The notion of "plan" is frequently referred to in the programming research literature. "Plan" is, e.g., the central concept of one of the main theoretical approaches to programming in psychology, the schema-based approach (cf. Détienne, 1990). In this context, the notion "programming plan" is generally used as a synonym of "programming schema", but Ormerod (1990) makes some remarks on the difference between the two notions which argue for preferring the term "schema" in this context (cf. also the distinction made by Rist, 1991).

"Programming plans" are supposed to be the representational structures expressing the knowledge possessed by programmers. Such as these plans have been identified, this knowledge is rather "static": programming plans tend to express the representation of programs rather than that of the programming activity. As noticed by Davies (1989): "Most [studies carried out in order to provide support for the notion of the programming plan] are concerned with an examination of what might be called a theory of 'programming plans' rather than with a theory of planning in programming. Empirical work has tended to address those questions that relate to what might be thought of as the static elements of plans ... rather than with the dynamic elements of plan based behaviour that are suggested by a theory of planning -in particular, questions that derive from a consideration of strategic differences in plan implementation." (p. 488)

This focus on static aspects of knowledge may be due to the experimental tasks traditionally used to study and identify programming knowledge structures. Indeed, even if one may claim that "the same kind of knowledge is used by the processes of program composition and program comprehension" (Détienne, 1990, pp. 205-206), research on "programming plans" has been mainly conducted in comprehension tasks. As Bisseret (1987/1990) puts it, "perhaps it has ... been too readily assumed, at a theoretical level, that if the comprehension activity could be explained, then an explanation of the production activity would follow automatically. In fact, production is quite different from comprehension, even if they have some cognitive processes in common, and more research should be undertaken into the production process." (p. 214)

2.2 Critiques to the central role of "plans" in cognitive activities

With respect to the critiques on the concept of "plan" in programming, the main reason leading to question its central role in the (expert) activity is the importance of strategical aspects the role of which is neglected by the plan-based approach (cf. the studies referred to in Gilmore, 1990).

Studying, next to program comprehension, also program generation, Davies (1989) has obtained interesting results in two experiments conducted on expert and intermediate programmers working

on programs in Pascal and in Basic (first experiment: a program generation task; second experiment: a program recall task). The only differences in the performances were strategic:

- the numbers of plan structures generated or recalled did not vary between experts and intermediates, nor between Pascal and Basic programs;

- what differed were the number of intra- and inter-plan jumps: experts made more inter-plan jumps whereas intermediates more intra-plan jumps. For the generation task, this meant that "the linear or serial generation strategy exhibited by intermediates is replaced [in the expert activity] by a strategy that focuses upon the iterative generation of code fragments that correspond to particular plan structures" (p. 497).

As Gilmore (1990) notices, the fact that "expert programmers have a much wider repertoire of strategies available to them as they program than do novices ... does not dispute that experts have 'plan-like' knowledge, but questions the importance of this knowledge itself versus an understanding of when or how to use that knowledge" (ibid., p. 223). So, Gilmore intends the "strategic approach ... as a complementary, rather than alternative, explanation of expertise" (p. 224). The plan-based theory falls short if (or when) it considers acquisition of expertise as exclusively, or even mainly, the acquisition of knowledge (implying "declarative", i.e. not covering the mastery and use of procedures and strategies). The strategies acquired by experts "may determine success more than does the programmer's available knowledge" (ibid., p. 233).

The concept of "plan" (implying "preexisting -") has also received more generally oriented critiques, e.g., in the "situated action" model by Suchman (1987/1990) and in the research of Chapman on "improvisation" in action (cf. Chapman, 1987).

The "situated action" model presented by Suchman (1987/1990) claims that the main guidance of purposeful action -i.e. of most action- does not come from plans: "Actions are primarily situated, and ... situated actions are essentially ad hoc" (p. ix), i.e. "taken in the context of particular, concrete circumstances" (p. viii). "Plans are best viewed as a weak resource for what is primarily ad hoc activity." (p. ix)

Action "is not predetermined, but ... neither random. A basic research goal for studies of situated action, therefore, is to explicate the relationships between the structures of action and the resources and constraints afforded by physical and social circumstances." (p. 179) Suchman judges that plans do not play an important role in actual action, i.e. purposeful action. She does -therefore?- not analyze the nature of these "negligible" entities. She claims that one reacts to specific "physical and social circumstances", but being an anthropologist, she does not ask, nor

examine, the "cognitive psychological circumstances", i.e. the knowledge and/or mental processes underlying these (re)actions.

Chapman's (1987) paper starts by describing TWEAK, a nonlinear, conjunctive domain-independent planner, and by presenting it as the A.I. solution to planning for conjunctive goals. Chapman considers that the main contribution of TWEAK is that it is "a simple, precise, implemented algorithm, which has been proved correct and complete". "Yet I wonder about the psychological reality of this sort of planning. It may be that the only solutions to the frame problem[2] we can devise are heuristic. Anecdotal evidence suggests that humans solve problems by improvisation, doing something easy and debugging the result when it fails (Suchman, 1985). Sussman's HACKER worked that way; unfortunately the set of bugs that it could patch are ones that TWEAK never introduces, and so his specific debugging techniques are of no use." (p. 365) Chapman announces that, with his colleague Agre, they have started more research on this improvisation and on planning as a "derivative activity", taking "situated activity" as the primary phenomenon. They are developing a theory of situated activity that does not involve planning.

3. PLANNING AND ORGANIZATION: EMPIRICAL STUDIES

An examination of the history of research on the organization of the activity in design tasks shows, globally, four stages: a first period in which the terms "hierarchy" or "opportunism" are not relevant, as they had not yet been used; a second period in which several studies concluded that design was organized in a hierarchical way; a third period when various authors started to analyze design as an opportunistically organized activity; a fourth period -in which we still are- in which several researchers have begun to qualify the "conflict" between "hierarchy" and "opportunism".

The position with respect to planning proposed in this paper, and which is developed below mainly for design activities (even if other types of activities might be concerned as well) is the following. The actions taken in a design activity cannot be predicted (completely) as stemming from preexisting plans; they are not systematic. The design actions which are taken into consideration for execution depend on the data which a designer has at the time: specifically, the state of their design in progress, their representation of this design and their knowledge, and the information which they have at their disposal and which they receive. The choice of a design action then depends on design action-selection criteria, the most important one of which is cognitive economy (see Visser, 1990).

---

[2] The frame problem, identified by McCarthy & Hayes (1969), is a "difficulty, common to all representation formalisms, ... their inability to model side effects of actions taken in the world by making corresponding modifications in the database representing the state of the world" (Barr & Feigenbaum, 1981, p. 177) (note W. Visser)

In order to defend this position, results from empirical studies conducted on design will be analyzed. These studies were carried out on various types of design domains (and one study on another type of ill-defined problem solving). Each of them shows one or more factors contributing to the opportunistic character of the organization of the activity. The studies will be presented according to the position the authors take on the "conflict" between "hierarchy" and "opportunism".

The claim defended in this paper requires that one takes into consideration a factor which has not been considered in most studies, i.e. the evaluation of possible design actions with respect to their "cognitive cost". This factor will be presented briefly in the last part of this section and will be discussed in more detail in Section 4.

3.1 Before the "conflict"

The authors of the two studies presented in this section did not focus especially on the "hierarchical" - "opportunism" dimension; the studies do, however, provide data relevant to this dimension.

3.1.1 Byrne (1977): design of meals

An early empirical design study is that by Byrne (1977) on the "planning of meals" which the author considers to be "a familiar and practiced task". Every subject can be considered an expert in this task. As in the study by Hayes-Roth and Hayes-Roth (1979) presented below, the "planning" studied by Byrne covers as well the global meal-design activity as its planning component.

The observed activity mainly seems to be organized hierarchically, and most decisions with respect to the choice of a course (a meal component) are made in a top-down fashion. However, some deviations of the *a priori* plan structures are observed. In spite of a preexisting "standard" plan for processing the subgoals of meal planning, Byrne observes "goal reordering", generally leading to an order which allows goal satisfaction without any risk of backtracking. Byrne considers this to be the "easiest" order (i.e. "[economizing] on effort").

Two concepts introduced by Byrne are of interest here: "Abstract Descriptions" and "Need slots". "Abstract Descriptions" represent the "goals which an adequate [meal] exemplar must satisfy" (p. 311). They could be identified with "plans" "for mentally constructing or mentally evaluating examples of [the corresponding classes] of meal" (p. 327).

"Need slots" allow the representation of information "whose lack would later make processing liable to disruption" (ibid.).

Byrne proposes that the "Abstract Description" of a three course meal would contain the three goals corresponding to these three courses, C1, C2 and C3, and that these goals are probably ordered: C1 then C2 then C3. On the other hand, he suggests that the goals C1 (and C3) being constrained by C2 be represented in a Need Slot, at such a location in the representation of C1 (and C3) that this constraint is encountered as soon as goal C1 (or goal C3) is attempted.

An example of the observed "goal reordering" is that C2 is generally processed before C1 and C3, i.e. the order actually adopted allows goal satisfaction without any risk of backtracking.

If a subject considers C1 (and C3) to be constrained by C2, one may wonder why the subject's "Abstract Description" of a three course meal does not contain the goals ordered as: C2 then C1 - C3. One explanation could be that the "standard" plan does not represent a procedural plan for the construction of a meal component, but a representation of the final result of the construction. So, next to its function in evaluating proposed components, this plan could serve as a declarative plan for the activity. Even if meal planning is "a familiar and practiced task" (Byrne, 1977, p. 287), subjects would not seem to possess a preexisting procedural plan for it. Subjects would only be able to construct such a plan through their practice of the activity in a great number of tasks executed consecutively (such as the six tasks executed in the experiment).

3.1.2 Malhotra, Thomas, Carroll and Miller (1980): restaurant design

The study by Malhotra et al. (1980) is marginally relevant here, because the authors examined designs for restaurants produced by subjects who were college students from a liberal arts college, i.e. novice designers. The authors qualified the subjects' final designs by two different scores: a score for "originality", i.e. the amount of information present in a subject's design compared to the maximal amount of information that could have been present in the design (looking across all designs produced by the different subjects), and a score for "practicality", i.e. the percentage of required goals satisfied.

The authors also collected data on the design activity using questionnaires asking designers questions on their plan(s) and strategies.

"Surprisingly", according to the authors, "a majority of the subjects claimed to have designed top-down, to have planned their approach, and to have tackled the more difficult problems first. However, none of these selfreported strategy variables were correlated with [the scores of their final designs]. What was predictive of [these scores] were the subject's expressed goals" in terms

of "having a design that was novel, imaginative and original" or rather "workable and practical" (p. 127).
The authors seem to suppose that there should be a correlation between (self-reported) strategies and the final results of an activity. This is however only a hypothesis, which would need to be examined on the basis of data concerning the supposed relationship.

3.2 Hierarchy

Especially in the domain of software design, a certain number of studies have been conducted in order to examine how (rather than "if" and, only if yes, "how") designers organize their activity hierarchically. In their presentation of results, the authors insist on the hierarchical aspects of design, but generally notice some "exceptions", presented as "details" with respect to the general hierarchical organization.

The three studies presented below are an example of this approach which consists in examining how designers organize their activity hierarchically. The first two of them are often referred to as "the" software design studies showing the hierarchically structured character of software design. The second study could indeed be considered as showing that even the design activity required by a nonroutine software design task may be organized in a completely hierarchical manner by an expert. The confrontation of the results of this study, however, with those from other research conducted by the same authors asks for the introduction of some nuances into this general conclusion.

3.2.1 Jeffries,Turner, Polson and Atwood (1981): design of a page-keyed indexer software system

Focusing on the control processes used in expert design, the authors formulate the hypothesis that it is the mastery of decomposition that differentiates experts and novices in their design activity. Thus, central in the authors' data analysis is then the question of how subjects decompose the problem they have to solve and/or how they decompose their global problem-solving task.

Jeffries et al. (1981) study four experts and five novices on a problem which is of "moderate difficulty" for the novices ("upper-division undergraduate" level); one may suppose that it is a rather easy problem for the experts.

The authors start their presentation of results asserting that "almost all subjects approached the problem with the same global control strategy", i.e. problem decomposition (p. 270). In the Discussion, they assert that "experts expand sub-problems systematically, typically top-down,

breadth-first" (p. 280). A close reading of the paper shows, however, that this "systematic" strategy has numerous "exceptions".

Only one expert out of four shows a systematic implementation of a top-down, breadth-first strategy. The other experts deviate more or less from this strategy. The authors present some examples of these "exceptions": a designer may choose to deviate from the "advocated order" when he realizes that a component has a known solution, is critical for success, or presents special difficulties. In our interpretation, the authors are providing here three factors contributing to an opportunistic organization of the activity.

Some examples of behaviors shown by the three experts who deviated from the "optimal" strategy are the following:

- descending in the decomposition tree, but coming back up afterwards, e.g., to introduce a whole new solution decomposition level;

- starting the decomposition by a top-down processing of only some branches of the tree or starting the decomposition in the middle of the tree;

- working simultaneously on two distinct branches;

- making interruptions for digressions at another than the current level, e.g., to handle other subproblems or to define primitive operations, i.e. elements at the lowest level.

If one wants to take into account these behaviors, one needs a model which allows for opportunism!

3.2.2 Adelson and Soloway (1988): design of an electronic mail system (EMS)

The interest of this study is that, in confrontation with a companion paper (presented below), it allows to relate the nature of the design activity to the characteristics of the combination "problem" and "subject".

In order to develop a model of software design, the authors want to observe "problem solving" and not the use of "routine cognitive skills". So they provide their subjects with a task "challenging to them". They are asked to solve a problem which was "similar to the types of problems [they] had to deal with professionally", but leading to a "novel" task, because "none of [them] had designed a solution to the problem previously".

Three expert software designers having previous relevant specialized experience with communication systems, but having never designed an EMS, are asked to design a simple EMS. All three experts implement a breadth-first strategy (called "balanced development" by the authors). Repeatedly, until the design is complete, they try to achieve their top level goal: Expand (Next level). "At any point in time, the resolved processing modules of the design were all defined at approximately the same level of detail." (p. 187)

Handling a problem at one level, a designer may think of related elements at another level. Sometimes, experts are capable of maintaining this kind of elements in memory and retrieving them at the appropriate moment. "Frequently", however, the experts were observed "[making] 'notes' to themselves about things to remember later in the design process .... [concerning] constraints, partial solutions, or potential inconsistencies" (p. 188). Adelson, Littman, Ehrlich, Black and Soloway (1985) posit the existence of "demons", "active information gatherers" which would remind the designer to incorporate such information into the design once the appropriate level of analysis has been reached. One may note the resemblance with the "Need slots" proposed by Byrne (1977).

So, this study shows that even in a nonroutine software design task, experts may organize their activity in a completely hierarchical way. Let us confront these results with those from other research conducted by the authors.

3.2.3 Adelson and Soloway (1985): design of an interrupt handler

In this study, the authors examine several design problems, varying in their degree of "familiarity to the observed designers". An interesting observation in the context of the present paper has been made on an expert who solved two problems. The first was the design of an EMS, which constituted for this designer a "mixed" problem with respect to familiarity in that it concerned an "unfamiliar object" in a "familiar domain". The second problem, the design of an interrupt handler, was globally "familiar" to him: it concerned a "familiar object" in a "familiar domain"; but it included an "unfamiliar" subproblem: the designer did not know the particular chip used as the interrupting device.

Once the designer had developed an abstract solution to the globally "familiar" interrupt handler problem, "he turned his attention to the functionality of the [unfamiliar subproblem of the] chip" and explored it in detail. This differed from the way he handled the EMS problem of "mixed" difficulty, where "exploration was cut off sooner and postponed via the making of notes".

The authors explain this difference as the designer "allowing" himself to deviate from "systematic expansion" when he is familiar with a problem domain. "If the mental model is lost from working

memory it can easily be reconstructed if it has been constructed frequently in the past. As a result, details can be explored ... and then the mental model can be reconstructed when the designer is ready to continue systematic expansion." (p. 1358)

3.3 Opportunism

This section presents six studies in which the authors conclude that design (or, in the case of the second study, another type of ill-defined problem solving) activities are organized opportunistically.

The first study is often used as "the" reference for the opportunistic nature of planning, considered here as the design of plans.

The second one is not a study of design. It is an example of another class of ill-defined problem solving activities. This study on the solving of problems in the domain of the social sciences is included in this presentation because it examines "ill-defined" problems which are different from design problems, but which share many characteristics with them. Its inclusion may contribute to making our conclusion more general.

The next three studies cover computational geometry algorithm design, mechanical design and traffic signal setting.

The last study presented in this section is the study by Guindon, Krasner and Curtis (1987) on the design of software to control lift movement. It is often referred to in the software design literature as showing the opportunistic nature of this type of design.

Already in 1980, Green, in a paper on "Planning a program", concludes a discussion of structured programming methods by advancing the idea that "good programmers .... leap intuitively ahead, from stepping stone to stepping stone, following a vision of the final program; and then they solidify, check, and construct a proper path. That proper path from first move to last move, in the correct order, is the program, their equivalent of the formal proof." (p. 306)

Green also notes that the author who introduced the concept of "stepwise refinement", Wirth, is himself "quite explicit; having described his stepwise refinement, [Wirth] says 'I should like to stress that we should not be led to infer that the actual program development proceeds in such a well organized, straightforward, top-down manner. Later refinement steps may show that earlier ones are inappropriate and must be reconsidered.'"(Green, 1980, p. 306)

Until the "Second Workshop on Empirical Studies of Programmers" in 1987, there had been, however, no empirical studies focusing on the opportunistic organization of the design activity. At

this Workshop, two papers were presented which questioned the traditional software design models (such as those by Jeffries et al., 1981, and Adelson et al., 1988, presented above, pleading for "hierarchy"). These were the Guindon et al. (1987) study and Visser (1987) which will be presented below. Both made proposals for other models: Guindon et al. (1987) in terms of "serendipitous" design, Visser in terms of "opportunism".

3.3.1 Hayes-Roth and Hayes-Roth (1979): design of errand plans

This famous study, serving as "the" reference concerning the opportunistic organization of design activities, has a tricky aspect. The "planning" examined by the authors was as well a component activity of the global activity, as the main activity itself (as in the Byrne, 1977, study). So the "errand planning" they examined may be considered as the activity of "designing a plan for the errands", where the resulting "plan" is a route connecting all errands. We will refer to the main activity studied by the authors (the errand planning) as "design" (of plans), reserving "planning" for the component activity. This allows us to distinguish the main activity and the planning component, and to compare the activity studied by the authors with the other design activities presented in this paper.

In the model elaborated by the authors, plan design is conceived as an incremental, opportunistic process (see also Hayes-Roth, Hayes-Roth, Rosenschein & Cammarata, 1979). A tentative "plan" is elaborated by several cooperating cognitive "specialists" (knowledge sources) making decisions concerning which they can interact and communicate via the "blackboard". This common data structure is partitioned into several planes containing conceptually different categories of decisions: one is the "plan" (next to the "executive", "meta-plan", "plan abstraction" and "knowledge base" planes).

The assumptions of this model, which generalizes the theoretical architecture of the Hearsay-II system, is illustrated with a subject's "thinking aloud" protocol. The authors show how the model can produce this protocol.

The sequences of the subject's basic actions, i.e. his decisions, included both top-down and bottom-up instances; plans were formed at low levels in the absence of corresponding higher-level plans: processing was multi-directional. The authors note that, compared with a fixed, one-directional -in particular, a top-down- planning approach, "the bottom-up component in multi-directional processing provides a potentially important source of innovation in planning" (p. 306). They refer to Feitelson and Stefik (1977) who made similar remarks concerning the "largely event driven nature of the planning process" observed by Feitelson and Stefik on an expert geneticist who is interpreted to proceed in this event-driven way with the aim of "fishing for interesting possibilities" (ibid.).

The plans designed by the subject observed by Hayes-Roth and Hayes-Roth were elaborated, not retrieved. This elaboration was not systematic: the subject proceeded neither in a strictly breadth-first, nor in a strictly depth-first way; he used neither only a top-down, nor only a bottom-up approach; his actions did not fit a simple hierarchical structure. His design grew "by incremental accretion": "each new decision [was related by the planner] to some subset of his previous decisions. ... The developing plan need not grow as a coherent integrated plan. Alternative subplans can develop independently either within or between levels of abstraction. The planner can incorporate these subplans into the final plan as she or he wishes." (p. 304)

3.3.2 Voss, Greene, Post and Penner (1983): political science problem solving

The class of ill-defined problems examined in this study comes from the domain of the social sciences. The authors observed experts, novices, graduate students in the domain of political sciences and "nonexpert experts" (i.e. advanced graduate students and political-science faculty members not specialized in the particular problem domain, i.e. Soviet Union agricultural problems).

In this study, "experts did not articulate their highest level plans" and, "during the initial phases of protocol generation, [they] showed no evidence of having a well-developed solution plan" (Voss et al., 1983, p. 191).

Subproblems were generated in two ways: they were either stated as part of a decomposition process or they were "encountered" when "the implications of a proposed solution [were explored]. ... This is an important distinction, for it is primarily the experts that generated subproblems via the second mechanism. ... Novice protocols are characterized by problem decomposition in which solutions are proposed for a number of relatively low-level subproblems." (p. 193) Graduate students and "nonexpert experts" formed a "transition" between novice and expert performance.

3.3.3 Kant (1985): computational geometry algorithm design

Kant (1985) and Kant and Newell (1984) collected protocols from several graduate and undergraduate students in computer science. The protocols of two subjects were analyzed in great detail, "while the others have been gone over more lightly and used primarily as confirming evidence" (Kant, 1985, p. 1362). In Kant and Newell (1984), the two designers on whose protocols the authors focus are qualified, the one being "moderately sophisticated in theory and algorithm design", the other being "fairly sophisticated in algorithm design" (pp. 100-101), but both are described as knowing little about the particular domain addressed in the problem used in the experiments, convex hulls. Besides, Kant (1985) considers that the complexity of the individual

algorithms to be designed is "due to a requirement for cleverness rather than to the information processing overload of combining an overwhelming number of small but straightforward parts" (p. 1362).

Kant (1985) proposes the concept of a "kernel idea" as playing a particularly important role in the observed activity: design generally starts by "selecting and sticking" with an idea which is "quickly selected from those known to the designer ... [who] lays out the basic steps of the chosen idea and follows through with it unless the approach proves completely unfeasible (p. 1362) (see the "primary generator" proposed by Darke, 1979/1984). Even if the elaboration of the kernel idea proceeds by stepwise refinement, this "process is hardly one of pure top-down design". Refinement is brought about by difficulties arising during problem-solution development (missing steps, inconsistencies between parts of the algorithm). In addition, "if one aspect of the algorithm is a potential problem ..., then it is more likely to be expanded to ensure that the algorithm as a whole is feasible" (p. 1366). "Often, the designers notice things about the sample figures that they were not looking for. When [this] turns out to be useful in developing their algorithm, [the author considers] that they have made a discovery.... [which] could be characterized as serendipitously satisfied goals." (p. 1364) Refinement is also guided by evaluation in the form of "algorithm execution [exposing] opportunities for improvement or modification of an algorithm" (p. 1363).

With respect to control, the author notices that "the design processes described ... do not always run to completion and do not take place in any fixed order. Evaluations ... may cause the designer to terminate one approach and go on to another. The ordering of the design processes (including when they begin and end) seems to arise naturally out of the mechanism of trial execution." (pp. 1365-1366)

Kant concludes: "In short, design processes are applied as appropriate. Control .... comes out of responding to the data and out of the problems and opportunities arising during execution." (p. 1366)

3.3.4 Ullman, Dietterich and Staufer (1988): mechanical design

In order to construct a model of mechanical design, Ullman et al. (1988) studied how two different types of design problems were solved by graduate students and experienced mechanical designers. The experienced subjects could be expected to have a high degree of expertise for the selected problems and it is on these subjects that the paper provides the greatest amount of information. The rest of this presentation will be devoted to them.

For these experienced designers, the authors come to a conclusion which is similar to that reached by Kant (1985) who studied rather non-routine design. "The design process is controlled locally, at

the episode and task level. The designer does not formulate and then execute a global design plan execution. Instead, a central concept is developed and gradually extended to accomplish the design goals" (Ullman et al., 1988, p. 36). This "central concept" may be compared with the "kernel idea" proposed by Kant (1985).

The authors notice that, firstly, designers progress from systematic to opportunistic behaviour as the design evolves and, secondly, they do not always keep their designs "balanced" (see Ullman, Staufer & Dietterich, 1987).

"Initially, ... subjects ... established an initial agenda for solving the problem. This was followed by a systematic period of conceptual design. The subjects usually followed an organized plan of attack at this early stage. As the design progresses, subjects became more opportunistic." (Ullman, Staufer & Dietterich, 1987, p. 64) Attention may switch from the current problem to "another (usually adjacent) problem [suddenly 'noticed']" (ibid.). "Ideas and problems are triggered by noticing patterns or configurations in the sketches. Sometimes, the focus of attention is immediately shifted to one of these, perhaps because the designer doesn't want to lose the 'good idea' by forgetting it. We also suspect that as the design progresses, the complexity increases to the point where the designer can only keep a small portion of the whole design in short term memory or in front of him or her in the form of a sketch. Consequently, an agenda is formed by what he or she can remember and see." (Ullman, Staufer & Dietterich, 1987, p. 65)

Little planning was observed (13% of the design "episodes"). The observed "plans were usually rather short-range, near-term plans. Their main purpose seems to be to evaluate whether a proposed task is worth performing at the current time. ... Plans, once formulated, were not followed very exactly.

Another hypothesis ... is that plans are formed prior to tasks in which many distractions are possible.... Such a plan may provide a kind of defense against the possible distracting effects" of information that is going to be encountered during the next episode(s) (Ullman, Dietterich & Staufer, 1988, p. 43).

Concerning "balanced development", the authors "hesitate to draw any conclusions from [the] observations because of the ambiguity of the term.... We define it as the effort to keep all elements of a design at the same level of abstraction while moving the design toward completion. [However] we do not yet have an objective definition of levels of abstraction" (Ullman, Staufer & Dietterich, 1987, p. 65).

3.3.5 Bisseret, Figeac-Létang and Falzon (1988): traffic signal setting

The authors studied eight experienced technicians solving "a rather difficult problem". They refer to the Hayes-Roth and Hayes-Roth (1979) study and the critical review presented by Carroll and Rosson (1985) of the "analytic approach" to design. This leads Bisseret et al. (1988) to formulate as the basic hypothesis of their study that "in [design] activities, resolution steps do not appear according to a stable, pre-planned strategy, but rather dynamically, according to an 'opportunistic' strategy" (p. 2). They consider the blackboard model to be "promising for the psychological modeling of these activities, and perhaps even for the modeling of problem solving in general" (ibid.).

The results of the detailed analysis of six subjects' verbal protocols "tend to give credit to the hypothesis ...: they are characteristic of an opportunistic mode of reasoning" (p. 4). "The six subjects produce six different solutions; the subjects do not announce an a priori resolution plan. The ways in which resolutions proceed ... differ largely, from both standpoints of their plans and of the nature of the successive steps. A common plan can only be found at a very high level of generality. ... As soon as a less abstract level is considered ..., the resolution steps widely differ." (ibid.)

The verbal protocols have been analyzed according to a blackboard model. Bisseret et al. (1988) construct a "knowledge base made of the gathering of the knowledge structures (rules, schemas, etc.) of all subjects" (p. 7). The authors give a "description of the subjects' reasoning in terms of a sequence of solution states (the content of the blackboard) and of solving steps" (ibid.). They have started the elaboration of a blackboard structure and present a simplified representation of it illustrated by a detailed example of one subject's reasoning episode.

3.3.6 Guindon, Krasner and Curtis (1987): design of the software to control lift movement

Guindon et al. (1987) studied eight professional programmers working on a lift problem which took them some two hours to solve. The problem presented "novelty because none of [the] designers had designed an identical system in the past, although they had worked on related real-time, concurrent, or embedded systems." (Guindon, 1990, p. 314)

The protocols of two designers were analyzed in detail (see also Guindon, 1990). The observed design activities were qualified as "serendipitous": even if the observed design behaviour is interspersed with top-down decomposition episodes, control of problem solving proceeded by recognition of partial solutions, at different levels of abstraction or detail, without the designers having previously decomposed the problem into subproblems.

This recognition of solutions to subproblems, often at another level of detail or abstraction than the currently handled problem, may be triggered by familiar aspects of the problem environment that happened to be focused on. Understanding and elaborating the requirements through mental simulations often led to the sudden discovery of new -added or inferred- requirements. This contributes to the "serendipity" when the corresponding solutions are developed immediately (rather than simply noted in order to be processed later on). Examination of external solution representations also led to recognition of solutions to subproblems before "their turn". The designers expanded their solutions by rapidly shifting between different levels of abstraction and different parts of the solution decomposition and by developing low-level partial solutions prior to a high-level decomposition. All the "unbalanced" design activities taken together made up 52% of the total of design activities (for the two designers under study in the analyses).

3.4 Hierarchy or opportunism?

The studies presented in this section come from what we consider, in the history of research on design organization, to be the period we are still in and in which research is trying to qualify the "conflict" between "hierarchy" and "opportunism".

Three positions are presented to this respect:

- Hierarchy or Opportunism is a question of level of analysis
- Hierarchy or Opportunism is (only) a question of action-execution knowledge
- Hierarchy or Opportunism is (also) a question of action-management knowledge

Before presenting the studies defending these positions, our general approach to problem-solving is briefly set out.

Modeling problem solving at two levels: action execution and action management. We distinguish the level of the actual design problem-solving actions (action execution) from the control level (action management) where decisions on the priorities of these problem-solving actions are taken. In a presentation of elements for a blackboard model of design, where the focus was on the control component, we proposed that these two levels be articulated according to the following iterative sequence (see Visser, 1990):

(i)	design knowledge:	action proposal
(ii)	control knowledge:	action evaluation and selection
(iii)	design knowledge:	action execution
(iv)	back to (i)	

At the execution level, design problem-solving actions are proposed and executed; at the control level, the action proposals are evaluated and one action is selected for execution. The analysis of the data obtained in Visser (1990) led us to propose "cognitive cost" as control's main action-selection criterion: if several action proposals are made, control will select the most "economical" action.

3.4.1 Hierarchy or opportunism: a question of level of analysis. Davies (1991): simple programming problems

In a paper entitled "Characterizing the program design activity: neither strictly top-down nor globally opportunistic", Davies (1991; see also Davies & Castell, to be published) concludes that an opportunistic organization holds only for local episodes in the activity implemented in design tasks; it does not hold for the global organization of the design activity. According to the author, "a top-down hierarchically leveled approach" is the main global design strategy adopted by experienced programmers. "While opportunistic episodes may occur at any point in the evolution of a program" - Davies observes that they appear rather as the task progresses than in early design stages- "the programming activity itself is hierarchically structured and proceeds in a largely top-down fashion" (Davies, 1991, p. 173).

This conclusion prompts us to make three remarks which will be developed in more detail below. First, the problems examined are rather simple (the most difficult one does not take more than 78 minutes to be solved by an experienced programmer/designer). The second remark, which constitutes a more important objection against the conclusion reached by the author, is that the top-down model may be considered as a special case of the opportunistic model, but not vice versa. Third, Davies concludes that "opportunistic episodes arise largely as a consequence of simple cognitive failures .... related to working memory capacity limitations" (p. 185). In our model, the limited capacity of working memory is only one of the various factors contributing to making a designer choose one action rather than another.

3.4.2 Hierarchy or opportunism: (only) a question of action-execution knowledge. Rist (1990): simple programming problems

Using, like Davies, (very) simple programming problems, which -as does Davies- he analyzes very precisely and in great detail, Rist seeks to qualify the recently "accepted" conclusion that design

activities are opportunistically organized. He examines how the nature of design activity -such as its organization- depends on the knowledge available to the designer. He concludes that designers' activities may be organized hierarchically if these designers know the solution to the problem they are solving, i.e. possess the corresponding schema. If they do not, their activity will be organized opportunistically.

"If the designer could retrieve a known solution [a plan schema] to the problem, the solution was applied and the program was designed forward from the input and output, as each part of the solution was expanded in turn. If a solution could not be retrieved, it had to be created by designing backward from the goals of the problem. More precisely, it was created by a process of backward and bottom-up design from an initial sketchy solution or focal idea, that was expanded to define a complete solution. Interaction between the two approaches created the complex behaviour observed in the study." (p. 307).

Rist notices that "experienced programmers coding a large program, or novices coding a small one, rarely have a complete plan in mind when they design a solution. ... [They use] an approach similar to the 'island driving' technique used in many blackboard systems ... , in which a solution is built up around a set of initial, partial attempts (islands) until the islands can be connected to form a complete solution." (p. 307)

3.4.3 Hierarchy or opportunism: (also) a question of action-management knowledge. Visser (1987, 1990): functional specifications design and software design (real tasks)

Even if the conclusions of Rist's study may seem to concur with the stand taken in this paper, there is an important difference.

Underlying our claim that, even for experts involved in routine design, retrieval of preexisting plans does not appropriately characterize the organization of their activity, is the idea that, in real design tasks, designers may often find it profitable to deviate from these plans. Indeed,

- even if designers can and, in fact, do retrieve a known solution plan for their problem (which may be possible in routine design, as shown by the Rist and other studies, and as we will see below in the Visser studies),
- yet if they evaluate the cost of their possible actions ("cognitive" and other costs), as they will do in real design,
- the action selected for execution will often be an action other than the one proposed by the plan.

It is not surprising that, like the authors of the studies presented in the previous sections, Rist did not take into account the action evaluation-and-selection. Indeed, this factor may be expected to make its appearance only in "real" design, where questions of "cost" acquire importance.

Two design studies will be used to defend our claim. They were carried out on professional, industrial designers involved in real design tasks, a characteristic of these studies which -as we will argue- is responsible for us coming to a conclusion different from the previously analyzed research. One study was on specifications design, the other on software design.

In the first study (Visser, 1990), an experienced mechanical engineer was observed, during a period of three weeks, throughout his definition of the functional specifications for the control part of a "programmable controller" (a computer specialized in the control of industrial processes, here a machining process). In the second study (Visser, 1987), a rather experienced software engineer, was observed, for four weeks full time, throughout his designing and coding of the software for the control part (whose specifications were those defined by the mechanical engineer). Both the software and the specification design tasks were globally "routine" tasks, but there were subtasks having "nonroutine" characteristics.
Visser (1991a, 1992) presents, through examples from these two and a third of her empirical design studies, an analysis of design at three levels: the global organizational, the strategic, and the problem-solving process level.

Visser (1990): functional specifications design. This first study focused on the possible differences between the designer's mental representation of the organization of his activity, and the actual organization of this activity. So we asked the designer to describe his activity, explaining that we were interested in the actual organization, not in a rationalization of it, such as a linearization or any other type of structuring. He presented us with a description most appropriately represented by a "hierarchically structured" plan[3]. So he possessed a complete solution schema for the specification problem.

This representation of his activity, i.e. a hierarchical action structure accompanied by a procedure which covers it, reflects a "procedural plan". The designer may have thought that he actually followed this plan, but our observations showed that he deviated from it whenever other possible actions, or local plans, that he perceived, were judged more "interesting" from a cognitive-economy viewpoint. The mental representational structure described by the designer was certainly a plan - even the main one- which guided his activity, but it was followed only as long as no possibilities arose for "cognitive-cost opportunities", i.e. for actions which were more economical. When they did, the designer deviated from his plan.

---

[3] The plan is said to be "hierarchically structured" and not "hierarchical", to avoid confusion with "hierarchical planning" as described by Sacerdoti (1974).

We identified both the processes which led to deviation-action proposals and "cognitive-cost opportunities" and the control criteria used in order to select, from these proposals, the action to be executed. They will be presented in the next section.

The difference between a designer's mental representation of the organization of his or her activity, and the actual organization of this activity may be related to Suchman's (1987/1990) remark about the "reconstruction" of plans when presented in retrospect. According to Suchman, "it is only when we are pressed to account for the rationality of our actions ... that we invoke the guidance of a plan. Stated in advance, plans are necessarily vague, insofar as they must accommodate the unforeseeable contingencies of particular situations. Reconstructed in retrospect, plans systematically filter out precisely the particularity of detail that characterizes situated actions, in favor of those aspects of the actions that can be seen to accord with the plan." (Suchman, 1987/1990, p. ix)

We believe however that designers have and use these plans which they describe, but that they are not the only guiding structures for the activity.

Visser (1987): design of the software for controlling an automatic machine tool. In the second study, the software designer was of course supposed to deliver a "correct" program. There was, however, an entire, independent "testing and debugging" stage following his intervention on the machine-tool project. This third "stage" was executed by another software engineer, and took place in the workshop, in parallel with the mechanical testing of the machine tool (Visser, 1988, globally presents the longitudinal three-stage study: the two studies presented here and the third one on the testing and debugging of the code, not discussed in the present paper).

We observed that the software designer, after one hour of planning, directly started to code (and continued for over four weeks). This coding was frequently interrupted for other activities (from planning to solution evaluation), but the interruptions were not systematic.

With respect to the organization of the activity, we tried to break down the software designer's activities into consecutive "stages". The distinction criterion adopted was the following: a sequence of actions is considered to constitute one "stage" when the actions are homogeneous with respect to their function (e.g., planning, problem understanding, solution development, solution evaluation) (see also Pennington & Grabowski, 1990).

Using the distinction criterion, a first and a second stage could be neatly identified:

- Problem representation. The first day, the software designer "studied" and "analyzed" the specifications he had received. Actually, he skimmed through the some 50 cm of A4 documents corresponding to these specifications.

- "Programming planning". During one hour, the software designer plans his programming activity. This leads to a "programming plan". The designer does this planning along two lines:

  - Breakdown of his task into program parts, according to the relative urgency with which various colleagues (especially in the workshop) need the different parts. This leads the designer to distinguish in the program three parts, which he plans to handle consecutively.

  - Breakdown of the program into modules, according to the different machine-tool functions. This decomposition follows the order of modules on the listing of an "example" program that the software designer had previously written for a similar machine tool.

After these first two stages (Problem representation and Programming planning), occupying all together one day, the designer directly started to code -and continued for over four weeks. This coding, however, was -of course, one should say- often interrupted. As well as coding, the designer carried out one or the other of the different types of activities distinguished above (planning, problem understanding, solution development, solution evaluation), which may be considered as components of a "design" activity. These different types of activities not necessarily occurred iteratively, in a fixed order. Thus there were no separate "stages" in the software designer's activity during the rest of the four weeks, neither a "design", nor a "coding" stage. The designer's activity during this period will be simply qualified as "programming".

So, except during the first short "programming planning" stage, the software designer's "planning" was "local" and "punctual". During the rest of the time, i.e. during the entire period in which he was "programming", it took place at varying problem-solving levels and concerned larger or smaller entities (e.g., at the design level, it could be a function or a machining operation; at the coding level, it could be a module or an instruction). The designer had a "programming plan" inspired by the order of the modules in the "example" program, that he reused heavily, but he deviated from this plan if another order was judged more economical from the point of view of cognitive cost. Local deviation actions, or alternative local plans leading to more global deviations, were triggered by various processes. These processes may be analyzed in the same terms as those proposed for modeling the deviations observed on the mechanical designer involved in his specification task.

4. OPPORTUNISTIC ORGANIZATION OF DESIGN: WHAT, WHY AND HOW?

The results presented above have led us to conclude that expert design activities, even in routine tasks, are opportunistically organized. In this section the main factors of opportunism are identified and elements are presented for a model of design which is able to take into account its opportunistic organization.

A short general characterization of "opportunism" will be followed by a discussion of two types of factors contributing to the possibly opportunistic organization of an activity: "situational" factors, characterizing "problem situations", and "processing" factors, qualifying a designer's knowledge sources and their use in design problem solving. "Situational" factors are considered to result from a combination of task factors and subject factors, so they depend upon the subject's knowledge and processing of the task. That is why we judge that the second type of factors, the "processing" factors, are the most relevant in a model of design activities.

4.1 Opportunism

The authors who introduced the concept of "opportunistic planning" in psychology, Hayes-Roth and Hayes-Roth (1979), consider the activity observed in their study, i.e. sequences of errand decisions, to be "fairly opportunistic" because "at each point in the process, the planner's current decisions and observations suggest various opportunities for plan development. The planner's subsequent decisions follow up on selected opportunities" (p. 276). "Each decision is motivated by one or two immediately preceding decisions, rather than by some high-level executive program" (Hayes-Roth, Hayes-Roth, Rosenschein & Cammarata, 1979, p. 381).

Our approach to "opportunism" is inspired by Hayes-Roth and colleagues. In their model, planning is under the control of the "executive", which "selects one of the invoked specialists to execute its action" (p. 291). We distinguish between action proposal and action selection, and consider an activity -here design- to be organized "opportunistically" if the actually executed design actions depend, at each moment t, on the evaluation of the actions proposed at t -and not on a pre-established plan which is followed without a possibly previous comparison with other action proposals.

Which actions are proposed at a moment t depends on the data which designers have at that moment: these data are mainly the designers' representation of the state of their design in progress (thus, the result of the preceding design actions), their knowledge (domain knowledge, design knowledge, more general problem-solving knowledge) and the information at their disposal and received from other sources (the client providing -and modifying- the requirements; colleagues making remarks and suggestions or providing other information; technical documents). In Hayes-

Roth and Hayes-Roth's experimental setting, the problem specifications which were given to the subjects did not receive any later addition or modification -as they do in real-life tasks. One may suppose that it is due to this setting that the authors did not identify the important role of other information than that stemming from a designer's previous actions and knowledge.

The evaluation of the proposed actions is based on action-selection criteria. The analysis of the data gathered on the specifications design task led us to propose "cognitive economy" as control's main action-selection criterion: if several action proposals are made, control will select the most economical action. Various factors contributing to an action's cognitive cost will be presented below, as will be the criteria proposed by Hayes-Roth and Hayes-Roth.

4.2 Problem situations: which characteristics can make a designer's activity opportunistically organized?

We claimed above that ultimately it is the evaluation of the proposed design actions which causes a designer's activity to become opportunistically organized. "Ultimately", because if all action proposals are made by a preexisting hierarchically structured plan, the activity will be organized hierarchically. Thus we join Hayes-Roth and Hayes-Roth (1979) in considering the top-down model as a special case of the opportunistic model.

This is also the conclusion of Bisseret et al. (1988) who used a blackboard architecture to model the cognitive activity involved in traffic signal setting (conceived by the authors as a design activity). They considered as "a very important advantage [of the blackboard architecture to model the cognitive activity] that, notwithstanding its opportunistic control basis, the model does not preclude an a priori defined control, but includes it as a special case" (Bisseret et al., 1988, p. 12).

Hayes-Roth and Hayes-Roth propose that "one resolution of the apparent conflict between the [successive refinement model and the opportunistic model] would simply incorporate the top-down model as a special case of the opportunistic model" (p. 307). The top-down problem-solving method "'define and refine' is only one of many problem-solving methods adoptable in the framework of the opportunistic model" (p. 308).

So, in order for an activity to be organized in an opportunistic way (i.e. possibly with hierarchical episodes at a local level, but not globally hierarchical), there are, in addition to a possible plan, other knowledge sources proposing design actions depending on the data the designer has at each moment t. These knowledge sources will be presented below as elements for a model of design. Before this presentation, we will set out how characteristics of problem situations may contribute to the way in which an activity is organized.

"Problem situations". Rather than characterizing a particular task as a "problem", independently of a subject (designer) or other cognitive system confronted with this task, we consider that a problem only exists relative to a subject (see Hoc, 1988b; Mayer, 1989). Defining "a situation [as] a functional system constituted by a task and a subject" (Leplat & Hoc, 1983, p. 49), a situation constitutes a "problem" -or not- for a particular subject depending on the representation this subject constructs of her or his task. For Hoc (1988b), a situation constitutes a "problem" for a particular subject if this subject does not have at his or her disposal a "legal" procedure to achieve the task goal. We follow Mayer (1989) (see his definitions presented above in Section 1), in considering a situation to constitute a "problem" for a subject if its representation constructed by the subject does not elicit a memorized answer, but if it must be solved by either applying a well-known procedure ("routine problems") or by generating a novel procedure ("nonroutine problems").

Given this approach to problems and problem solving, most of the characteristics which, in the problem solving literature, are considered to be "problem characteristics" might be considered to be related to the characteristics of the combination problem and subject confronted with the corresponding task.

4.2.1 "Problem characteristics"

Hayes-Roth and Hayes-Roth (1979) argue that, "[because the top-down method] 'define and refine' is only one of many problem-solving methods adoptable in the framework of the opportunistic model, ... the question is no longer which model is correct, but rather, under what circumstances do planners bring alternative problem-solving methods to bear?" (p. 308). As an answer to this question, the authors suggest "three variables which might influence a planner's approach to a particular problem: problem characteristics, individual differences, and expertise" (p. 306). "Problem characteristics" is discussed below; "individual differences" and "expertise" in the subsection on "subject characteristics".

Problem structure. With respect to the "problem characteristics", Hayes-Roth and Hayes-Roth (1979) assert that "Planners might usefully exploit a top-down approach to planning whenever the problem at hand exhibited an inherent hierarchical structure" (p. 306). The authors present as an example of a "problem exhibiting an inherent hierarchical structure" the menu planning problem used by Byrne (1977), whereas their own errand-planning problem "did not exhibit any obvious hierarchical structure" (p. 307). The reasoning followed by Hayes-Roth and Hayes-Roth seems to presuppose that the "structure" of a problem is the structure of its final solution -something not especially evident to us. If it were the case, the results obtained in our specifications design study (Visser, 1990) are contrary to the conjectured relationship between "problem structure" and a planner's preferred (or most appropriate) "planning approach". The final design to be constructed

by the mechanical engineer observed by Visser -a functional specifications schema- had a neat hierarchical structure. So did the "plan" the designer said to follow -and which he was indeed observed to follow, but not systematically: the organization of his activity was opportunistic.

Rist (1991) relates the structure of a problem and the expertise of the designer solving the problem. "Expertise determines whether a schema can be retrieved, but the problem determines how easy it is to apply. When the structure of the solution is not apparent, top-down design will break down, and some other design method then has to be used." (p. 250)

Problem structuredness. Guindon (1990) claims the "structuredness" of a problem to be an important variable with respect to the approach taken by a subject who has to solve the problem. She identifies three factors as contributing to this problem "structuredness": the degree of "novelty" of the problem, the degree of completeness of its specifications and the number of knowledge domains to be used in solving it.

"Supporting [the argument that the degree of 'novelty' of a problem is related to its 'structuredness'], Designer 2, who had developed systems more similar to the [problem used in the study] than had Designer 1, exhibited a design process that matched more closely a top-down approach than did Designer 1. Thus, ... design problems that are simple or that present little novelty can be solved by and large in a top-down manner." (p. 335)

Problem presentation. Carroll, Thomas, Miller and Friedman (1980), in a study of the design of a library system, examined the influence of the structure of the explicit problem presentation on four dependent variables: solution time, satisfaction of design requirements, stability of solution process (i.e. the sequence of problem-solving steps or stages), and "clustering" (i.e. the degree to which the solutions reflect the inherent problem structure). They showed that problem presentations which were explicitly more hierarchically structured resulted in more stable design protocols and more clustered solutions.

The well-known influence of the wording of a problem statement (see Mayer, 1989) has been examined by Siddiqi and Ratcliff (1989) by varying the wording (but, nota bene, together with the content) of the specifications for a program to be designed. They conclude that the bias they observed, in previous experiments, towards simplistic decomposition "not only can be counteracted, but also can be reversed with appropriate cues in the specifications" corresponding to "relatively minor changes ... in specification wording or content" (p. 401).

The reader may notice that, first, a change of content leading to another problem representation and, thus, to another solution approach does not seem very surprising; second, that the problem

specifications, in a real task, are only one of the various sources of information designers have at their disposal -and thus of the "problem presentation".

Time allotted for the task. A variable which has been studied by Hayes-Roth (1979, quoted in Hayes-Roth and Hayes-Roth, 1979) which can indeed be considered to be an "objective" "problem characteristic" is the "amount of time available for plan execution". Hayes-Roth (1979) "successfully induced alternative planning approaches by manipulating [this] amount of time. ... For problems that imposed severe time constraints, most subjects adopted a top-down approach. For problems that imposed minimal time constraints, most subjects adopted a bottom-up approach" (Hayes-Roth & Hayes-Roth, 1979, p. 307).

This observation may be related to the one made by Adelson and Soloway (1985) on a designer "allowing" himself to deviate from "systematic expansion" when he is familiar with a problem domain.

Delay of implementation. Voss et al. (1983) consider this variable as one of the two factors making the solving of social science problems quite difficult (the other factor being "the lack of agreed-upon solutions", discussed below). The fact that "in most social science problem solving there is a relatively long delay from the time a solution is proposed and accepted to when it is fully implemented" has "profound effects" on the solving of the problem (Voss et al., 1983, p. 169). This delay especially affects solution evaluation. "Naturally, a good solution anticipates changes in conditions, but anticipation can be quite difficult." (ibid.)

As argued above, we judge it more appropriate to consider that most of the five "problem characteristics" presented above are characteristics of a situation, i.e. of the couple problem-subjects (see also Brooks, 1977, suggesting that designers may proceed top-down if they solve a "problem" which is "familiar" to them and do so in a programming language they are experienced in). For most real design problems, it is difficult -if not impossible- to determine their "structure" (or "structuredness") independently of the designers solving the problems, especially the representation they construct of the problems, based on their knowledge and other aspects of their expertise. The same remark applies to "problem characteristics" such as "novelty of the problem" and "number of knowledge domains to be used in solving the design problem".

The argument that a problem cannot be characterized independently of its solver may be supposed to hold for all types of problems. For design and other ill-structured problems it may, in addition, be particularly difficult to define their "inherent" structure because the identification of the "structure" of these problems is precisely one of the main aspects of their solving (cf. also the remark by Ullman et al., 1987, concerning the difficulty to give an "objective definition of levels of abstraction").

4.2.2 Subject characteristics

In the problem-solving literature, the "subjects" variable has been examined mostly from the viewpoint of differences in expertise between subjects. Implicitly these differences have generally been considered as differences in level of expertise (expert vs. novice); differences in type of expertise have rarely been studied (but see Visser & Falzon, to be published, who have compared the way in which two expert designers in the same domain but with different design experiences differ with respect to various aspects of their knowledge organization; see also Falzon & Visser, 1989).

Next to levels of expertise, two other aspects of inter-subjects differences will be presented below, "individual differences" (implying "- other than due to expertise") and "inter-solvers agreement".

Expertise. Whereas in the problem-solving literature, expertise has been analyzed as depending on various factors (see Glaser, 1986; Glaser & Chi, 1988; Holyoak, in press), in studies on design most authors have focused on designers' knowledge structures. In the studies on software design these structures have mainly been formalized in terms of "plans" and/or "schemas". In section 2.2, we presented the main critiques this approach has received and we referred to several studies showing the importance of strategic differences between experts and novices.

The studies which focused on differences in design expertise in terms of knowledge structures, have shown generally that these differences led novice and expert designers confronted with a same design task to adopt different strategies: compared with novices, experts tend to adopt a "depth-first" approach, whereas novices use rather a "breadth-first" strategy.

Individual differences. This variable has been examined in a study by Hayes-Roth (1979; quoted in Hayes-Roth & Hayes-Roth, 1979). "Many of [Hayes-Roth's] subjects exhibited a strong proclivity to adopt a bottom-up approach regardless of problem characteristics. Even with explicit instruction, some subjects persisted in using the bottom-up approach." (Hayes-Roth & Hayes-Roth, 1979, p. 307)

Inter-solvers agreement. Voss et al. (1983) compare the social science problems used in their study with those "used thus far in problem-solving research". They argue that, while all these problems have one or more known solutions, the specificity of problems in the domain of the social sciences is their "lack of agreed-upon solutions": "social science problems seldom have solutions about which experts are in complete agreement" (p. 169). Voss and Post (1988) relate this point made by Voss et al. (1983) to Reitman's (1965) analysis of the role of agreement in constraint satisfaction among the community of solvers. In Reitman's terms, there is disagreement in the domain of social sciences regarding how to fill open constraints. According to Voss and Post, "it is

possible, of course, ... to have ill-structured physics problems. Such problems would be expected to occur, for example, in new areas of research." (p. 263) The authors refer to Tweney (1981) for evidence suggesting that, in these conditions, the solving of physics problems highly resembles that of political science problems.

Analyzing the composition of a fugue, Reitman was one of the first authors to conduct an empirical study of design problems, and to characterize ill-defined problems (cf. also Reitman, 1964). He considers that there is a continuum ranging from well-defined formal problems to such ill-defined empirical problems as composing a fugue. This continuum is related to the idea of "ambiguity", i.e. there is little or no agreement over a specified community of problem-solvers regarding the referents of the problem attributes, permissible operations, and their consequences. This leads to solutions to ill-defined problems which are more or less accepted, rather than correct (Reitman, 1964).

It is in a discussion of the "task environment" in social science problem solving that Voss et al. (1983) present two other examples of situational characteristics, i.e. depending on task and subject. They notice how the solution proposed to a political science problem may depend upon the professional status of the solver and to whom the solution is enunciated. The authors consider their observations to "indicate how the surroundings in which the problem is presented may provide constraints and influence what the individual states. Indeed, there apparently are 'audience effects' in social science problem solving just as in writing and speaking." (p. 209)

4.3 Problem processing: why and how does a designer's activity become opportunistically organized?

We suggested above that, in order for an activity to be organized opportunistically, there must be, next to a possible plan, other action-proposing knowledge sources. The selection among the possibly various action proposals was supposed to be made by the control using a number of action-selection criteria.

After a short description of these selection criteria, these action-proposing knowledge sources leading to "cognitive-cost opportunities" will be presented.

4.3.1 Cognitive economy: the main control action-selection criterion

The main goal underlying the designers' "choice" for the organization of their activity is supposed to be "cognitive economy" (Visser, 1990; see also Byrne, 1977). At each step in the design process, control selects one action from those proposed. If an alternative-to-the-plan action is more

profitable than the planned action from this cognitive-economy viewpoint, designers abandon - generally only temporarily- their plan.

The cognitive cost of an action is the cost of accessing the required information and of its processing in order to achieve the goal of the action. So an example of a factor contributing to the cognitive cost of an action is the availability of the particular information or of its schema for executing the action. Another factor is the degree to which the cues in the problem representation (constructed by the designer) lead to a direct access to the required information, or ask for elaboration (or "analysis") of the problem representation in order to create the appropriate cues.

A second action-selection criterion is the "importance" of the proposed design actions. We have suggested that this criterion only plays a role when the designer has to choose between two, or more, actions at equal cost. Examples of two factors contributing to the importance of an action are the importance of the type of the action and the importance of the type of object concerned by the action (for more details, see Visser, 1990).

Observations made by several authors may suggest still other factors contributing to the selection of an action. Kant (1985) observes that the most difficult, most problematic problem-solution aspects are handled first, while Spohrer (quoted in Rist, 1990) suggests that either "simplest-first" or "hardest-first" orders are possible. Rist observes (or only supposes?) that these "more rational methods" are reserved to experts and that, before acquiring this expertise, novices tend to proceed by random selection.

Guindon, Curtis and Krasner (1987) propose certain general heuristics which, once a subproblem or a set of subproblems is identified, will help determine whether to focus on that subproblem or which subproblem to select:

- *select most difficult subproblem first;*
- if the solution of a subproblem affects, or is necessary for, the solution for another subproblem, attempt to solve the former subproblem first;
- if a subproblem is at a too low level of detail, postpone trying to solve it;
- if a subproblem is known to be trivial, or at least easy in the sense that simple solutions can be retrieved directly from memory, postpone its solution.

A strategy affecting when focus on a subproblem shifts is the temporary abandonment of a subproblem for which no satisfactory solution has been reached after a certain amount of effort.

Davies (1991) considers "opportunistic episodes" to be the consequence of cognitive failures related to working memory capacity limitations. We suppose that this limited capacity does not

favour preferentially an opportunistic or a top-down approach. Plans which -if followed- would lead to a top-down approach may be interesting from a cognitive-economy viewpoint because they organize in a relatively small number of chunks (more or less manageable in terms of working memory capacity limitations) the information-processing elements necessary to execute the design actions. Alternative action proposals -leading, if the action is selected, to an opportunistically organized activity- may also be interesting from a cognitive-economy viewpoint: in this case the "gain" of plan deviation is to be compared with its "cost", i.e. the cost of resuming the plan. Resuming the plan may be relatively inexpensive because it requires plan retrieval from long-term memory. The gain of plan deviation may be relatively high because what constitutes a "cognitive-cost opportunity" is precisely an action which, if executed now, costs less than if executed later, when it would be "its turn" according to the plan (various examples may be found in Visser, 1990).

In the Hayes-Roth and Hayes-Roth (1979) opportunistic model, control of the planning process is done by one of the five blackboard planes, the "executive", which "selects one of the invoked specialists to execute its action" (p. 291). The "executive" plane has three levels. At the "priorities" level, decisions are taken about the allocation of cognitive resources during the planning process. "'Focus' decisions indicate what kind of decision to make at a specific point .... Finally, 'schedule' decisions resolve any remaining conflicts between competing specialists ... on the basis of relative efficiency, reliability, etc." (p. 290) Unfortunately, the paper provides no examples of schedule decisions in the analyzed subject protocol, so that the "relative efficiency, reliability, etc." is not made more concrete.

From an A.I. viewpoint, the term of "cognitive economy" is used by Lenat, Hayes-Roth and Klahr (1979) to describe a system's heightened productivity, without the sacrifice of expressibility. "*Cognitive economy is the degree to which a program is adapted to its environment*, the extent to which its internal capabilities (structures and processes) accurately and efficiently reflect its environmental niche." (p. 3) The authors propose a certain number of techniques enabling programs "to (semi-)automatically improve themselves and thus increase their productivity". The techniques of "caching" are contrasted with psychological ideas of economy and the psychological evidence presented is considered by the authors to support their idea.

"Caching" is "storing the results of frequently-requested searches, so [that] they needn't be repeated over and over again; i.e., *intelligent redundancy*" (p. 3). The authors refer to psychological experiments by Rips, Shoben and Smith (1973) and by Hayes-Roth and Hayes-Roth (1975, 1977). These authors presented counterexamples to the hierarchical retrieval schemes proposed by earlier cognitive psychological research on semantic networks (particularly, Collins and Quillian, 1969, 1972). Hayes-Roth and Hayes-Roth (1975) proposed "an adaptive model of memory, in which all learned relations are represented directly, with strength proportional to experience. There is also good evidence for redundancy in the network, with multiple routes

connecting nodes representing particular pairs of concepts." (Hayes-Roth & Hayes-Roth, 1975, p. 519, quoted in Lenat et al., 1979, p. 28) One may think also of the concept of "knowledge compilation" introduced by Anderson (1986) as "the general learning mechanism".

"Caching" techniques are the only ones which the authors discuss making reference to psychological data. The other techniques and characteristics proposed by Lenat et al. (1979) as being useful for intelligent systems "having more potentially interesting things to do than they have resources to pursue" (p. 41) could, however, also be psychologically "plausible". These are techniques for "dynamic self-monitoring and self-modification" (i.e. learning), "expectation-filtering", and the use of "multiple levels of abstraction". All these are characteristics which the authors suppose to be needed by "A.I. systems ... addressing the same question facing intelligent humans: 'What would I most like to accomplish next, and how can I do that economically?'" (p. 42).

4.3.2 Cognitive processes for proposing actions leading to an opportunistically organized activity

According to the definition of "opportunism" presented above, the main factors leading to alternative-to-the-plan design actions -and thus possibly to an opportunistically organized activity- stem from "taking advantage of" the data considered by the designer, i.e. from processing of information, and of (permanent) knowledge and (temporary) design representations (cf. the role of "interesting possibilities" suggested by Hayes-Roth and Hayes-Roth, 1979, and that of "discoveries" and "good ideas" proposed by Kant, 1985). Reformulating the six processes presented in Visser (1990, where more details and examples can be found), we identified five possible candidates for being "taken advantage of":

- Information designers are "presented with":
  - by the client;
  - by a colleague;
  - "by themselves" when "drifting":

  Different types of "drifting" may be distinguished, according to
  (a) the problem-solving stage in which drifting occurs, and
  (b) the source providing the information from which it occurs. It may lead to phenomena described subjectively as "thinking of X", or "rather doing Y now" because one is afraid of "forgetting it" otherwise.
  (a) Drifting may occur
  · when the designer is looking for information required for the current design action, thus before he has identified this information; or
  · once he has this information, but before he has used it for the current design action.

(b) The designer may drift from information considered on an external information source, or from data processed in memory (knowledge).

In all these cases, the designer comes to focus on information other than that required for the current design action.

- Information used for the current design action when considered from another viewpoint (e.g., for a designer involved in functional specification, information used for this functional specification task considered also, or rather, from its mechanical, i.e. physical, viewpoint).

- Information "constructed" (or "obtained") by
  - analyzing the problem specifications;
  - developing and evaluating solutions;
  - exploring the implications of a proposed solution

  (see Guindon, 1990; Kant, 1985; Voss et al., 1983).

- Mental representations of design objects activated through activation-guiding relationships existing between them and the representations used for the current design actions. These relationships (such as analogy, prerequisites, opposites and interaction) between mental representations may lead to switches between the design-object representations which are under focus.

  The "goal reordering" observed by Byrne (1977) may be related to this process. In order to avoid the need for backtracking, the designer may process design components which are constrained by other components in such an order that the corresponding goals be satisfied without the risk of backtracking. In order to achieve this "optimal" "goal order", the design component representations must be related in memory and their relationships must be perceived and exploited by the designer.

- Mental representations of design procedures which may be available because they have just been developed.

  The action proposal of taking advantage of such an available procedure may be selected by the control because it satisfies the "cognitive economy" criterion: retrieving a procedure as one single chunk from memory is less expensive than having to develop it from pieces (later on, when it will no longer be available as a single chunk).

Taking advantage of these "opportunities" rather than following a pre-established plan will lead to an opportunistically organized activity.

N.B. Not following a plan is not necessarily due to plan deviation. A designer may follow a plan, but skip an action proposed by this plan. This may occur when the planned design action is judged

as being too "difficult" (see, e.g., Jeffries et al., 1981; Kant, 1985; Ullman et al.,1987) or, more generally, when the planned action P is judged as costing too much, not compared to a currently proposed deviation action, but to P itself if executed later. This may occur, e.g., when the procedure for obtaining the information required by the planned action is judged as being too expensive (e.g., because the information is not available or is very expensive to obtain now). In this case, P is postponed and a local plan is generally formed for re-proposing P, either as soon as the conditions leading to its postponement no longer prevail, or at a particular moment later in the design process.

5. DISCUSSION

Before closing with a discussion of some of the consequences of our conclusions for design support tools, this section will sum up the main points of this paper.

This paper has presented three different hypotheses on the relationship between the routine character of a task (subsuming the dimension of "expertise" of the designer confronted with the task) and the organization of the corresponding activity. Visser (to be published) has summed these up as follows:

- **the "naive" hypothesis: routine design tasks have a systematic organization**, or, to be more precise, the activity implemented in a task is organized systematically as long as the task has a routine character (cf. the observations by Davies, 1991, and by Ullman et al., 1988, that designers progress from systematic to opportunistic behaviour as the design evolves);

- **the "realistic" hypothesis based on experience in the laboratory: nonroutine design tasks have a systematic organization**, or, to be more precise, the activity implemented in a task becomes organized systematically as soon as the task becomes nonroutine (cf. the suggestion by Adelson and Soloway, 1985, that designers may "allow" themselves to deviate from "systematic expansion" only when they know a problem domain very well; cf. the hypothesis formulated by Ullman et al., 1988, that plans are formed prior to tasks in which many distractions are possible, as a defense against these possible distractions; cf. the observation made by Hayes-Roth, 1979, quoted in Hayes-Roth and Hayes-Roth, 1979, that only when time constraints are not severe, do designers tend to adopt a bottom-up approach);

- **the "perverse" hypothesis based on experience in the "real world": real design tasks have an opportunistic organization**, because the activity implemented in

a task can be organized systematically as long as the task has a routine character (plan retrieval is possible) and can be handled systematically once the task becomes nonroutine (for reasons of "cognitive protection"), but in real tasks the activity will generally not be systematic because of constraints at the action-management level (especially cognitive economy) (cf. the studies by Visser).

The results discussed in Section 3 were considered as a plea for the third hypothesis: there is more to expert knowledge than just planning. Preexisting plans which -if they are followed- may lead to systematically organized activities, are supposed to be only one of various action-proposing knowledge structures. The actions proposed are compared because only one action can be selected for execution. The main selection criterion is an action's "cognitive cost". Preexisting plans may be interesting from a cognitive-economy viewpoint because executing an action for which such a schematic memory representation is available may cost relatively little if all schema variables relevant for execution have constant or default values. But if other knowledge structures propose relatively more economical actions, designers may deviate from their plan. This is especially true for experts, who may be supposed to possess -or else to be able to construct without great difficulty- a representation of their activity which allows them to resume their plan later on, when it once again becomes profitable to do so from the viewpoint of cognitive economy.

Having to compare several action proposals and taking into account the cognitive cost of an action are two task characteristics which probably only make appear in "real" design. This may explain why in many laboratory experiments systematically organized design activities were often observed.

Section 4 presented two types of factors contributing to the possibly opportunistic organization of an activity: "situational" factors, characterizing "problem situations" (i.e. task-subject couples), and "processing" factors, qualifying a designer's knowledge sources and their use in design problem solving. Because "situational" factors depend upon a designer's knowledge and processing of the design task, we judged that the "processing" factors are the most relevant ones for a model of design activities. We briefly presented several elements for a model which is able to take into account the opportunistic organization of these activities.

Design support tools: assisting opportunistically organized activities? The conclusions of this paper may inspire, at least, two approaches to design assistance: given the opportunistic organization of the activity of designers working "in total freedom", one may either try to use tools to prevent such a way of proceeding, or offer designers such tools as would assist them in their "natural" way of proceeding.

So questions which may be asked are, e.g., the following. Should a designer be allowed to have an opportunistically organized activity? If so, should the designer be assisted in this activity -or should

assistance be restricted to systematical aspects of the activity, at local levels or even only at the global level? If one wishes to assist the designer (also) in the opportunistic aspects of the activity, how might this be done?

In an evaluation study of a programming environment which supports a particular top-down approach (a retrospective strategy), but in which plan revision is made very tedious, Hoc (1988a) shows that professional programmers working with this system may generate non-optimal solutions when, in order to repair planning errors, they awkwardly modify modules at a very low level.

Lebahar (1989) observes that the use of CAD in architectural design modifies the actual activity by inducing a hierarchical organization in the design solutions according to the "producible" objects, i.e. depending on the possibilities CAD makes available to its users.

We judge that assistance tools should be compatible with the actual activity. If design is opportunistically organized, a support system which supposes -and therefore imposes- a hierarchically structured design process will at the very least constrain designers and will probably even handicap them (see Visser & Hoc, 1990).

Most existing tools are, however, based on prescriptive or analytical, task-experience based models of design, i.e. not on data on the actual activity to be supported, and they are limited to certain tasks in the "detailed design" stage, e.g., to the "editing" task in text or program production, or to "drawing" in many engineering tasks.

In a paper focusing on the planning component in the production of text (not "editing" but "writing"), Bisseret (1987/1990) distinguishes between the plan of the product (the final text) which is hierarchical, and the plan of the activity (in our paper referred to as its "organization") which is opportunistic. He notices that most planning tools are plan *editors* which are based on a strictly hierarchical top-down model, and which, consequently, do not really support the planning process. He presents some research on tools which would not impose such a hierarchical planning structure.

Hoc (1988b) makes similar remarks with respect to assistance tools for software engineering, which are mostly program editors.

The problem, however, is not only one of developing an appropriate tool for supporting the use of a given language; it is also -or rather- a problem of the programming language as an appropriate tool. Green (1990) notices that "Unfortunately for [designers], not all devices make it possible [to use some version of opportunistic planning]. Some programming languages, for instance, pretty well forbid it. So - what do people do? Do they build their programs top-down, as they are

instructed? No. They abandon the programming language instead. They use informal languages which allow them to explore ideas. They develop their thoughts opportunistically, and then they rationalise and rebuild those thoughts in a coherent, well-structured way." (p. 2)

The empirical studies presented in this paper -and others which may be found in the literature- might lead to several suggestions for systems which offer "real" support. Such systems should

- not impose a hierarchical planning structure;
- allow designers, if they are following a plan, to deviate from -and even abandon- this plan;
- assist plan resumption, but not presuppose -and still less impose- it;
- keep a trace of abandoned, interrupted or postponed problems;
- assist representational activities:
  - allow designers to work at different levels of representation (different levels of detail, of abstraction, of design decomposition) and support such approaches;
  - make available to designers representations adapted to them according to their level of expertise;
  - make available to designers representations adapted to different viewpoints they may take on the design problem/solution state and assist their switching between these various viewpoints;
  - allow easy changes of representation mode (of a problem, of a solution);
- offer displays for helping the management of memory limitations, such as facilities for:
  - presenting intra- or inter-level information in parallel;
  - presenting constraints on the solution order;
  - maintaining a trace of abandoned, interrupted or postponed subproblems which may require backtracking;
- facilitate re-use (and/or analogical reasoning, especially access to analogs) (see also Falzon & Visser, 1989, 1991a, 1992).

More ideas on design assistance and references to other studies may be found in Ullman, Staufer & Dietterich (1987), Visser & Hoc (1990) and Whitefield (1986).

If there is more to design expert knowledge than is dreamed of in the current planner's philosophy, we may hope hope that this philosophy will soon broaden, leading to the development of tools which implement (at least some of) these suggestions!

REFERENCES

Adelson, B., Littman, D., Ehrlich, K., Black, J., and Soloway, E. Novice-expert differences in software design. In: B. Shackel (Ed.), Human-computer interaction - INTERACT '84. Amsterdam: North-Holland, 1985.

Adelson, B. & Soloway, E. The role of domain experience in software design. IEEE Transactions on Software Engineering, 1985, SE-11, 1351-1360.

Adelson, B., & Soloway, E. A model of software design. In M. T. H. Chi, R. Glaser & M. J. Farr (Eds.), The nature of expertise. Hillsdale, N.J.: Erlbaum, 1988.

Allen, J., Hendler, J., & Tate, A. Readings in planning. San Mateo, CA: Morgan Kaufmann, 1990.

Anderson, J. R. Knowledge compilation: the general learning mechanism. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), Machine learning. An artificial intelligence approach (Vol. II). Los Altos, Calif.: Morgan Kaufmann Publishers, 1986.

Barr, A., & Feigenbaum, E.A. (Eds.). The handbook of Artificial Intelligence (Vol.1). London: Pitman, 1981.

Bisseret, A. Towards computer-aided text production. In P. Falzon (Ed.), Cognitive ergonomics. Understanding, learning and designing human-computer interaction. London: Academic Press, 1990. (Reprinted from Research report N° 665, INRIA, Rocquencourt, 1987)

Bisseret, A., Figeac-Létang, C., & Falzon, P. Modeling opportunistic reasonings: the cognitive activity of traffic signal setting technicians (Research Report N° 898). Rocquencourt: INRIA, 1988.

Brooks, R. Towards a theory of the cognitive processes in computer programming. International Journal of Man-Machine Studies, 1977, 9, 737-751.

Brown, D. C., & Chandrasekaran, B. Design problem solving. Knowledge structures and control strategies. London: Pitman, 1989.

Byrne, R. Planning meals: Problem-solving on a real data-base. Cognition, 1977, 5, 287-332.

Carroll, J. M. & Rosson, M. B. Usability specifications as a tool in iterative development. In H. Rex Hartson (Ed.), Advances in human-computer interaction (Vol. 1). Norwood, N.J.: Ablex, 1985.

Carroll, J. M., Thomas, J. C., Miller, L. A. & Friedman, H. P. Aspects of structure in design problem solving. American Journal of Psychology, 1980, 93, 269-284.

Chapman, D. Planning for conjunctive goals. Artificial Intelligence, 1987, 32, 333-377.

Cheng, P. C-H. Modelling experiments in scientific discovery. Proceedings of the 12th International Joint Conference on Artificial Intelligence (Volume 2), August 1991, 739-744.

Darke, J. The primary generator and the design process. In N. Cross (Ed.), Developments in design methodology. Chichester: Wiley, 1984. (Originally published in Design Studies, 1979, 1, 36-44)

Davies, S. P. Skill levels and strategic differences in plan comprehension and implementation in programming. In A. Sutcliffe & L. Macaulay, L. (Eds.), People and computers V. Cambridge: Cambridge University Press, 1989.

Davies, S. P. Characterizing the program design activity: neither strictly top-down nor globally opportunistic. Behaviour & Information Technology, 1991, 10, 173-190.

Davies, S. P., & Castell, A. M. From individuals to groups through artifacts: the changing semantics of design in software development. In D. Gilmore, R. Winder & F. Détienne (Eds.), User-centred requirements for software engineering environments. Heidelberg: Springer, to be published.

Détienne, F. Expert programming knowledge: a schema-based approach. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming. London: Academic Press, 1990.

Falzon, P., & Visser, W. Variations in expertise: implications for the design of assistance systems. In G. Salvendy & M. Smith (Eds.), Designing and using human-computer interfaces and knowledge based systems. Amsterdam: Elsevier, 1989.

Gero, J. Foreword to Navinchandra, 1991.

Gilmore, D. J. Expert programming knowledge: a strategic approach. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming. London: Academic Press, 1990.

Glaser, R. On the nature of expertise. In F. Klix & H. Hagendorf (Eds.), Human memory and cognitive performances. Amsterdam: North-Holland, 1986.

Glaser, R., & Chi, M. T. H. Overview. In M. T. H. Chi, R. Glaser & M. J. Farr (Eds.), The nature of expertise. Hillsdale, N.J.: Laurence Erlbaum Associates, 1988.

Green, T. Programming as a cognitive activity. In H. T. Smith & T. R. G. Green (Eds.)., Human interaction with computers. London: Academic Press, 1980.

Green, T. Models of structure and models of perceiving structure (draft version). Proceedings of ECCE-5, Fifth European Conference of Cognitive ergonomics, Urbino (Italy), September 3-6, 1990.

Guindon, R. Designing the design process: exploiting opportunistic thoughts. Human-Computer Interaction, 1990, 5, 305-344.

Guindon, R., Curtis, B., & Krasner, H. A model of cognitive processes in software design: An analysis of breakdowns in early design activities by individuals (MCC Technical Report N° STP-283-87). Austin, TX: MCC, 1987.

Guindon, R., Krasner, H., & Curtis, B. Breakdowns and processes during the early activities of software design by professionals. In G. Olson, S. Sheppard & E. Soloway (Eds.), Empirical Studies of Programmers: Second Workshop. Norwood, N.J.: Ablex, 1987.

Hayes-Roth, B., & Hayes-Roth, F. A cognitive model of planning. Cognitive Science, 1979, 3, 275-310.

Hayes-Roth, B., & Hayes-Roth, F., Rosenschein, S., & Cammarata, S. Modeling planning as an incremental, opportunistic process. Proceedings of the 6th International Joint Conference on Artificial Intelligence, Tokyo, 20-08-1979.

Hoc, J. M. Towards effective computer aids to planning in computer programming. Theoretical concern and empirical evidence drawn from assessment of a prototype. In: G. C. van der Veer, T. R. G. Green, J. M. Hoc, and D. Murray (Eds.), Working with computers: theory versus outcomes. London: Academic Press, 1988a.

Hoc, J.M. Cognitive psychology of planning. London: Academic Press, 1988b.

Holyoak, K. J. Symbolic connectionnism: Toward third-generation theories of expertise. In K. A. Ericsson & J. Smith (Eds.), Toward a general theory of expertise: Prospects and limits. Cambridge, MA.: Cambridge University Press, in press.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. The processes involved in designing software. In J.R. Anderson (Ed.), Cognitive skills and their acquisition. Hillsdale, N.J.: Erlbaum, 1981.

Kant, E. Understanding and automating algorithm design. IEEE Transactions on Software Engineering, 1985, SE-11, 1361-1374.

Kant, E., & Newell, A. Problem solving techniques for the design of algorithms. Information Processing & Management, 1984, 20, 97-118.

Kaplan, C. A., & Simon, H. A. In search of insight. Cognitive Psychology, 1990, 22, 374-419.

Kulkarni, D., & Simon, H. A. The processes of scientific discovery: The strategy of experimentation. Cognitive Science, 1988, 12, 139-175.

Lebahar, J.C. La dialectique "compétence humaine/compétence artificielle" dans les processus de conception aidée par ordinateur. Contributions écrites - Premières journées de Psychologie du travail, Ergonomie et Psychopathologie du travail, PIRTTEM - CNRS. Paris, 13 et 14 juin 1989.

Lenat, D. B., Hayes-Roth, F., & Klahr, P. Cognitive economy (Working paper HPP-79-15). Stanford: Stanford University, Computer Science Department, Heuristic Programming Project, 1979.

Leplat, J., & Hoc, J.M. Tâche et activité dans l'analyse psychologique des situations. Cahiers de Psychologie Cognitive, 1983, 3, 49-63.

McCarthy, J., & Hayes, P. J. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie & B. Meltzer (Eds.), Machine intelligence 4. Edinburgh: Edinburgh University Press, 1969.

Malhotra, A., Thomas, J. C., Carroll, J. M. & Miller, L. A. Cognitive processes in design. International Journal of Man-Machine Studies, 1980, 12, 119-140.

Mayer, R. E. Human nonadversary problem solving. In K. J. Gilhooly (Ed.), Human and machine problem solving. New York: Plenum, 1989.

Miller, G. A., Galanter, E., & Pribram, K. H. Plans and the structure of behavior. London: Holt, Rinehart & Winston, 1960.

Navinchandra, D. Exploration and innovation in design. Towards a computational model. New York: Springer, 1991.

Ormerod, T. Human cognition and programming. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming. London: Academic Press, 1990.

Pennington, N. & Grabowski, B. The tasks of programming. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming. London: Academic Press, 1990.

Qin, Y., & Simon, H. A. Laboratory replication of scientific discovery processes. Cognitive Science, 1990, 14, 281-312.

Reitman, W. Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelley & G. L. Bryan (Eds.), Human judgments and optimality. New York: Wiley, 1964.

Rist, R. S. Variability in program design: the interaction of process with knowledge. International Journal of Man-Machine Studies, 1990, 33, 305-322.

Rist, R. Models of routine and non-routine design in the domain of programming. In J. S. Gero & F. Sudweeks (Eds.), Artificial Intelligence in Design (Preprints of the Workshop of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, 25 August 1991). Sydney: University of Sydney, 1991.

Sacerdoti, E. Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 1974, 5, 115-135.

Siddiqi, J. I. A., & Ratcliff, B. Specification influences in program design. International Journal of Man-Machine Studies, 1989, 31, 393-404.

Suchman, L. A. Plans and situated actions. Cambridge: Cambridge University Press, 1990. (Originally published 1987)

Ullman, D., Dietterich, T. G., & Staufer, L. A. A model of the mechanical design process based on empirical data. AI EDAM, 1988, 2, 33-52.

Ullman, D., Staufer, L. A., & Dietterich, T. G. Toward expert CAD. Computers in Mechanical Engineering, 1987, 6 (3), 56-70.

Visser, W. Strategies in programming programmable controllers: a field study on a professional programmer. In G. Olson, S. Sheppard & E. Soloway (Eds.), Empirical Studies of Programmers: Second Workshop. Norwood, N.J.: Ablex, 1987.

Visser, W. Giving up a hierarchical plan in a design activity (Research Report N° 814). Rocquencourt: INRIA, 1988.

Visser, W. More or less following a plan during design: opportunistic deviations in specification. International Journal of Man-Machine Studies, 1990, 33, 247-278.

Visser, W. The cognitive psychology viewpoint on design: examples from empirical studies. In J. Gero (Ed.), Artificial Intelligence in Design '91. Oxford: Butterworth-Heinemann, 1991a.

Visser, W. Evocation and elaboration of solutions: Different types of problem-solving actions. An empirical study on the design of an aerospace artifact. In T. Kohonen & F. Fogelman-Soulié (Eds.), COGNITIVA 90. At the crossroads of Artificial Intelligence, Cognitive science, and Neuroscience. Proceedings of the third COGNITIVA symposium. Amsterdam: Elsevier, 1991b.

Visser, W. Designers' activities examined at three levels: organization, strategies & problem-solving. Knowledge-Based Systems, 1992, 5 (1), 92-104.

Visser, W. Planning and organization in expert design activities. In D. Gilmore, R. Winder & F. Détienne (Eds.), User-centred requirements for software engineering environments. Heidelberg: Springer, to be published.

Visser, W., & Falzon, P. Catégorisation et types d'expertise. Une étude empirique dans le domaine de la conception industrielle. INTELLECTICA, to be published.

Visser, W., & Hoc, J.M. Expert software design strategies. In J.M. Hoc, T. Green, R. Samurçay & D. Gilmore (Eds.), Psychology of programming. London: Academic Press, 1990.

Voss, J. F., Greene, T. R., Post, T. A., & Penner, B. C. Problem-solving skill in the social sciences. In G. Bower (Ed.), The psychology of learning and motivation (Vol. 17). New York: Academic Press, 1983.

Voss, J. F., & Post, T. A. On the solving of ill-structured problems. In M. T. H. Chi, R. Glaser & M. J. Farr (Eds.), The nature of expertise. Hillsdale, N.J.: Laurence Erlbaum Associates, 1988.

Whitefield, A. An analysis and comparison of knowledge use in designing with and without CAD. In A. Smith (Ed.), Knowledge engineering and computer modelling in CAD. Proceedings of CAD86, Seventh International Conference on the Computer as a Design Tool. London: Butterworths, 1986.

ISSN 0249-6399